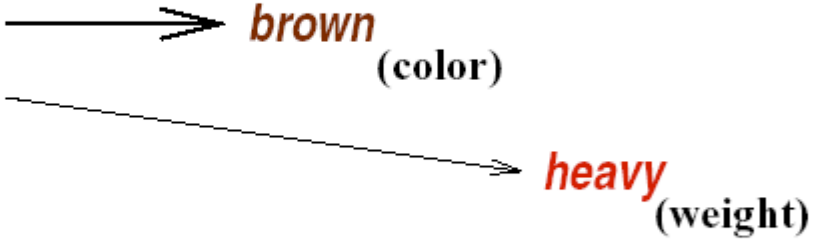
Interpretazione Astratta (cenni)

Agostino Cortesi

Astrazione:
selezionare una proprieta’

Astrazione:
selezionare una (delle) proprieta’



Astrazione e correttezza

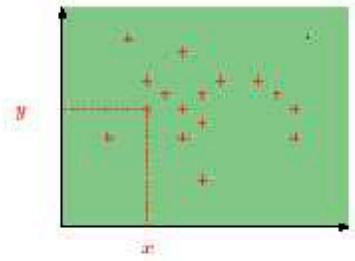
brown (color)

heavy (weight)

$\sqcup|$

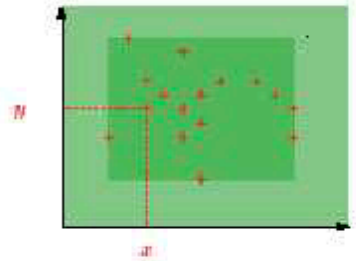
4000..6000 kg.

Astrarre un insiemi di punti nel piano…



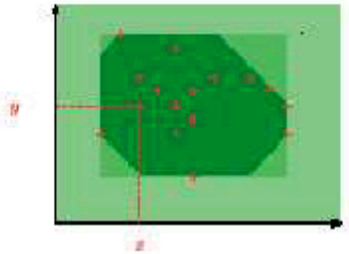
$$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$$

Fig. 2
SIGNS



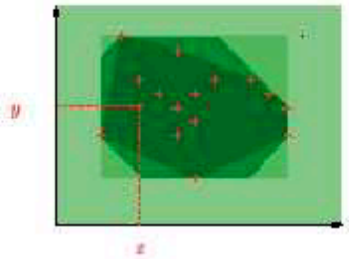
$$\begin{cases} x \in [3,\ 27] \\ y \in [4,\ 32] \end{cases}$$

Fig. 3
INTERVALS



$$\begin{cases} 3 \leq x \leq 27 \\ x + y \leq 88 \\ 4 \leq y \leq 32 \\ x - y \leq 61 \end{cases}$$

Fig. 4
OCTAGONS



$$\begin{cases} 7x + 31y \leq 325 \\ 21x + 7y \geq 0 \end{cases}$$

Fig. 5
POLYHEDRA

Interpretazione Astratta

- Una tecnica utilizzata da 30 anni (Patrick e Radhia Cousot, 1977) per trattare in modo sistematico astrazioni e approssimazioni
- Nata per descrivere analisi statiche di programmi imperativi e provarne la correttezza.
- Sviluppata per varie classi di linguaggi di programmazione e sistemi reattivi
- Vista oggi come tecnica generale per ragionare su semantiche a diversi livelli di astrazione
- Applicata con successo a sistemi distribuiti per verifica di programmi (correttezza – sicurezza)

L’idea generale

- il punto di partenza è la semantica concreta, ovvero una funzione che assegna significati ai comandi di un programma in un dominio fissato di computazione.
- un dominio astratto, che modella alcune proprietà delle computazioni concrete, tralasciando la rimanente informazione (dominio di computazione astratto)
- derivare una semantica astratta, che permetta di “eseguire astrattamente” il programma sul dominio astratto per calcolare la proprietà che il dominio astratto modella.
- il calcolo della semantica astratta tipicamente e` un calcolo di punto fisso
- sarà inoltre possibile calcolare una approssimazione corretta della semantica astratta

Semantica concreta

- Considereremo un linguaggio peudo-funzionale di base piuttosto che un linguaggio imperativo nello stile **While**
- Iniziamo da un linguaggio molto limitato, che permette unicamente di operare su moltiplicazioni di interi.

  Exp e ::= n | e*e
- La semantica di questo linguaggio si può descrivere mediante una funzione η definita da:

  η : **Exp** -> Z

  $\eta(n) = n$

  $\eta(e_1 * e_2) = \eta(e_1) * \eta(e_2)$

Semantica astratta

- Possiamo considerare un'astrazione della semantica concreta (semantica astratta) che calcola solo il segno delle espressioni

σ: **Exp** -> {-,0,+}

$$\sigma(n) = \begin{cases} - & \text{se } n<0 \\ 0 & \text{se } n=0 \\ + & \text{se } n>0 \end{cases}$$

$$\sigma(e_1 * e_2) = \sigma(e_1) *^a \sigma(e_2)$$

$*^a$	-	0	+
-	+	0	-
0	0	0	0
+	-	0	+

Correttezza

- Possiamo dimostrare che questa astrazione è corretta, ovvero che prevede correttamente il segno delle espressioni.

- La dimostrazione è per induzione strutturale sull'espressione e, e semplicemente utilizza le proprietà della moltiplicazione tra interi (il prodotto di due positivi è positivo, etc.).

Per ogni expressione e in **Exp**:

$\eta(e) < 0 \;,\; \sigma(e) = -$

$\eta(e) = 0 \;,\; \sigma(e) = 0$

$\eta(e) > 0 \;,\; \sigma(e) = +$

Una prospettiva diversa

- Possiamo associare ad ogni valore astratto l'insieme di valori concreti che esso rappresenta:

γ:{-,0,+} -> P(Z)

$\gamma(-) = \{x \in Z \mid x<0\}$

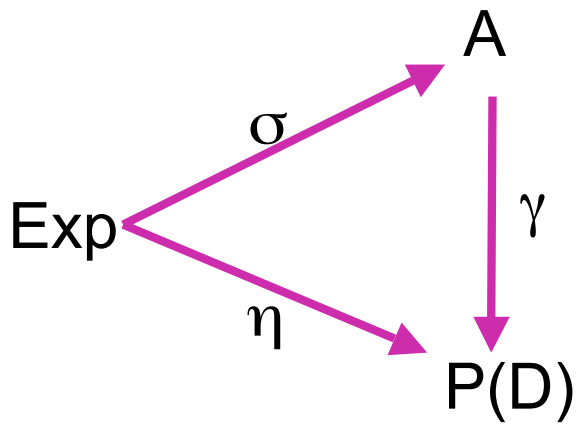
$\gamma(0) = \{0\}$

$\gamma(+) = \{x \in Z \mid x>0\}$

Concretizzazione

- La funzione di concretizzazione γ mappa un valore astratto in un insieme di valori concreti
- Indichiamo con D il dominio concreto dei valori e con A il dominio astratto

$$\eta(e) \in \gamma(\sigma(e))$$



Interpretazione Astratta

- Abbiamo specificato una interpretazione astratta.
  - Computazioni astratte in un dominio astratto
  - In questo caso, il dominio astratto è {+,0,-}.

- La semantica astratta è corretta
  - è un'approssimazione della semantica concreta

$$\{\eta(e)\} \subseteq \gamma(\sigma(e))$$

- La funzione di concretizzazione stabilisce la relazione tra il concetto di approssimazione nei due domini concreto ed astratto

Aggiungiamo -

- Aggiungiamo al nostro *tiny language* l'operatore unario di cambiamento di segno

  Exp e ::= n | e*e | -e

  $\eta(-e) = -\eta(e)$

  $\sigma(-e) = -^a\sigma(e)$ dove $-^a(-) = +$, $-^a(0) = 0$, $-^a(+) = -$

Aggiungiamo +

- Aggiungere l'addizione è più complesso, in quanto il dominio astratto non è chiuso rispetto a questa operazione

$$\eta(e_1 + e_2) = \eta(e_1) + \eta(e_2)$$

$$\sigma(e_1 + e_2) = \sigma(e_1) +^a \sigma(e_2)$$

$+^a$	-	0	+
-	-	-	?
0	-	0	+
+	?	+	+

- A quale valore astratto corrisponde il risultato della somma di due numeri interi con segno opposto?

Soluzione

- Aggiungiamo un nuovo valore astratto > che rappresenta un qualsiasi numero intero

$\gamma(>) = Z$

$+^a$	-	0	+	>
-	-	-	>	>
0	-	0	+	>
+	>	+	+	>
>	>	>	>	>

Estendere le altre operazioni

- Avendo aggiunto un elemento al dominio astratto, è necessario estendere le operazioni astratte già definite

$-^a$	-	0	+	>
-	+	0	-	>
0	0	0	0	0
+	-	0	+	>
>	>	0	>	>

$-^a(-) = +,\ -^a(0) = 0,$
$-^a(+) = -,\ -^a(>) = >$

Esempi

- In alcuni casi c'è perdita di informazione dovuta alle operazioni

  η((1+2) - 3) = 0
  σ((1+2)+ -3) = (+ +ª +) +ª - = + +ª - = >

- In altri casi non c'è perdita di informazione

  η((5*4) + 6) = 26
  σ((5*4) + 6) = (+ £ª +) +ª + = + +ª + = +

Aggiungiamo la divisione

- Aggiungere la divisione intera (con resto) / non da problemi, eccetto il caso della divisione per 0.
- Se dividiamo un insieme di interi per 0 che risultato otteniamo? L'insieme vuoto. Quindi la semantica concreta assumera` i propri valori sul powerset di Z, cioe` η:**Exp** -> P(Z)
- L'insieme vuoto di interi è rappresentato da un nuovo elemento $\perp$ astratto rispetto al quale si devono estendere le altre operazioni

$\gamma(\perp) = \varnothing;$

$/^a$	-	0	+	>	$\perp$
-	+	0	-	>	$\perp$
0	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
+	-	0	+	>	$\perp$
>	>	0	>	>	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

$\perp +^a x = \perp = x +^a \perp$

$\perp *^a x = \perp = x *^a \perp$

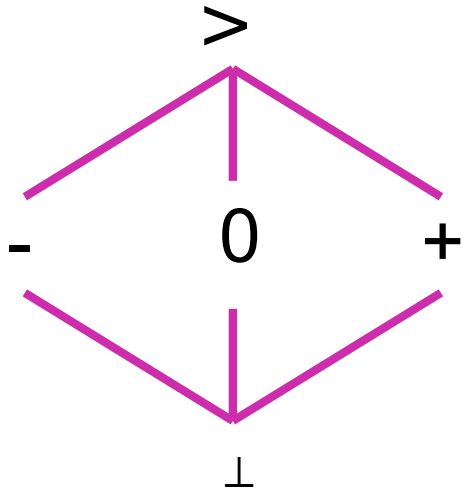
$-^a(\perp) = \perp$

Il dominio astratto

- Il dominio astratto è un poset in cui l'ordine parziale rappresenta la nozione di approssimazione/precisione
- L'ordine parziale è coerente con la funzione di concretizzazione:

$$x \leq y \;,\; \gamma(x) \subseteq \gamma(y)$$

- Ogni sottoinsieme ha un lub ed un glb: è quindi un reticolo completo



La funzione di astrazione

- Alla funzione di concretizzazione γ corrisponde una funzione di astrazione α.
- La funzione α mappa un insiemi S di valori concreti nel piu` preciso valore astratto che rappresenta S.
- Nel nostro esempio

$$\alpha : P(Z) \rightarrow A \qquad \alpha(S) = \begin{cases} \bot & \text{se } S = \emptyset; \\ - & \text{se } S \neq \emptyset,\ S \subseteq Z_{<0} \\ 0 & \text{se } S = \{0\} \\ + & \text{se } S \neq \emptyset,\ S \subseteq Z_{>0} \\ > & \text{altrimenti} \end{cases}$$

Definizione Generale

- Una Interpretazione Astratta consiste in:
  - Un dominio astratto A ed un dominio concreto C
  - A e C reticoli completi. L'ordine riflette la precisione/approssimazione (più piccolo = più preciso)
  - Funzioni di concretizzazione e di astrazione monotone, che formano una inserzione di Galois.
  - Operazioni astratte che astraggono correttamente su A la semantica concreta su C.

- Inserzione di Galois: funzioni monotone α:C -> A e γ:A -> C tali che:

  $\forall\, c \in C.\ \ c <_C \gamma(\alpha(c))$

  $\forall\, a \in A.\ a = \alpha(\gamma(a))$

Connessione di Galois

(α, C, A, γ) GC (Galois connection) se

1) A e C poset
2) α:C -> A monotona (astrazione)
3) γ : A -> C monotona (concretizzazione)
4) per ogni c in C. $c <_C \gamma(\alpha(c))$
5) per ogni a in A. $\alpha(\gamma(a)) <_A a$

Inserzione di Galois

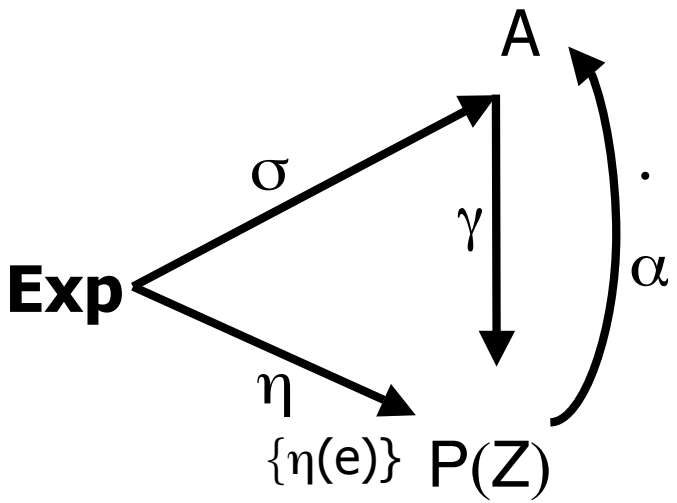
(α, C, A, γ) GC e` una Galois insertion (GI) se vale una delle seguenti condizioni equivalenti:

1) α surgettiva
2) γ iniettiva
3) per ogni a in A. $\alpha(\gamma(a)) = a$

Astrazione e Concretizzazione

- In una interpretazione astratta ci aspettiamo che il seguente diagramma commuti:

$$\eta(e) <= \gamma(\sigma(e))$$
$$\alpha(\{\eta(e)\}) \cdot \sigma(e)$$



Correttezza

- Per la correttezza dell'analisi sono necessarie le seguenti condizioni
- (α,C,A,γ) e` una GI
- Le operazioni astratte op^a (che sono supposte monotone) sono corrette rispetto alle corrispondenti operazioni concrete op (che sono pure supposte monotone): per ogni $(a_1,...,a_n)$ in A^n

  $op(\gamma(a_1),...,\gamma(a_n)) <_C \gamma(op^a(a_1),...,op^a(a_n))$
- La correttezza di un'operazione astratta op^a puo` essere equivalentemente definita come: per ogni $(c_1,...,c_n)$ in C^n

  $\alpha(op(c_1,...,c_n)) <_A op^a(\alpha(c_1),...,\alpha(c_n))$

Migliore approssimazione

- La condizione di correttezza garantisce quindi che il risultato dell'applicazione dell'operazione astratta sia una corretta approssimazione del risultato dell'applicazione della corrispondente operazione concreta.
- Per ogni operazione concreta op, possiamo sempre definire la cosiddetta migliore approssimmazione corretta di op sul dominio astratto A.
- $op^A(a_1,...,a_n)$, $\alpha(op(\gamma(a_1),...,\gamma(a_n))$

Semantica astratta

- Usando il dominio astratto possiamo definire una semantica astratta (di punto fisso) in modo analogo a quanto fatto per la semantica concreta
- Opportune condizioni di correttezza garantiscono che la semantica astratta e' una approssimazione corretta di quella concreta
- Opportune condizioni sul dominio astratto garantiscono che la semantica concreta e' computabile in tempo finito

- La semantica astratta ci permette di verificare (in modo parziale) proprieta' di correttezza dei programmi